AN INVESTIGATION INTO MULTI-CLOUD DATA PROCESSING FOR BIG DATA AS A SERVICE (BDaaS)

by

THALITA VERGILIO

A proposal submitted in partial fulfilment of the requirements of Leeds Beckett University for the degree of Doctor of Philosophy

Leeds Beckett University

School of Computing, Creative Technologies and Engineering

2017

CONTENTS

Abstract 4

Introduction 5

- Research Context 5
- Scope 6

Rationale, Aim and Objectives 7

- Rationale 7
- Research Questions 7
- Aim 8
- Objectives 8

Related Work and Definitions 10

- Related Work 10
- Definitions 14

Methodology 21

- Philosophical Approach 21
- Prototype Development 23
- Evaluation 24

Strategy 31

- Research Plan 31

Bibliography 38

FIGURES

Figure 1. Big Data Architectural Layers 6

Figure 2. Service Provider Managed Hybrid Cloud ("Cisco Intercloud Fabric: Hybrid Cloud with Choice, Consistency, Control and Compliance", 2016).....10

Figure 3. Proposed BDaaS Processing Layer 11

Figure 4. Cloud Dataflow provides a unified computation model for batch and streaming processing (Schmidt, 2015) 13

Figure 5. Processing paradigms (Casado and Younas, 2015) 16

Figure 6. PhD Gantt Chart 31

Figure 7. PhD Timeline 31

Figure 8. Year 1 32

Figure 9. Year 2 33

Figure 10. Year 3 34

Figure 11. Year 4 35

Figure 12. Year 5 36

Figure 13. Year 6 37

TABLES

Table 1. Risk Analysis.....30

ABSTRACT

Big data is an area of technological research which has been receiving increased attention in recent years. As the Internet of Things (IoT) expands to different spheres of human life, a large volume of structured, semi-structured and unstructured data is generated at very high velocity. To derive value from big data, businesses and organisations need to detect patterns and trends in historical data. They also need to receive, process and analyse real-time data in real-time, or close to real-time, a challenge which current technologies and traditional system architectures find difficult to meet. This research aims to investigate such challenges with a view to proposing a workable solution.

A number of architectures have emerged to answer distinct challenges posed by big data such as distributed batch processing of historical data, or the processing of streaming data in real-time. There is no single accepted solution to cater for all types of big data processing, so different technologies tend to be used in combination. Consequently, the learning curve for a developer working with big data is steep, and the processing logic developed within one system is generally not compatible with other systems, leading to code duplication and low maintainability. This research proposes to investigate the requirements for a unifying processing component for big data, and to create a microservices architecture where the processing logic encapsulated within each service component can be reused.

The experimental part of this research will involve the creation of a prototype to demonstrate the proposed microservices architecture. This prototype will be evaluated technically, via benchmarks, and empirically, through a case study followed by qualitative data analysis.

INTRODUCTION

RESEARCH CONTEXT

The advent and expansion of the IoT (Internet of Things) and has brought big data to the forefront of technology research. Smartphones, tablets, GPS trackers, sensors, video surveillance devices all produce a vast amount of data, in different formats, in real time. This data presents challenges not only in terms of storage, but also when it comes to processing and analysing it to extract information which is valuable to businesses and government agencies. A recent report published by CISCO shows that devices and connections are growing faster than the global population, which is partly due to the increase in the number of connected devices per household ("The Zettabyte Era—Trends and Analysis - Cisco," 2016). Over the next 20 years, this expansion in the industrial IoT is estimated to add $10 to $15 trillion to the global GDP, according to a report published by GE (Evans & Annunziata, 2012).

Microservices are a recent architectural pattern which emerged from challenges presented by large monolithical applications to real-world companies (Nadareishvili et al., 2016, p. ix). They are highly suited to the IoT, as they share the same requirements and the architectural goal of creating applications through distributed service composition (Butzin et al., 2016). A number of cloud service providers such as Amazon Web Services (AWS), Google Cloud Platform and Microsoft Azure have started to offer big data services for processing batch and stream data. However, there is very little integration between services offered by different cloud providers (Chen, H. et al., 2016). Moreover, collaboration between developers working with these technologies with a view to sharing requirements and reusing code is limited (Park et al., 2015).

One of the main obstacles to collaboration between developers working with Big Data as a Service (BDaaS) is the use of proprietary technology by cloud service providers. This means services built within one provider's platform will not necessarily work if transposed to a different provider's platform (Chen, H. et al., 2016). In order to achieve platform decoupling, an in-depth study of the common abstractions shared by BDaaS platforms' underlying processing languages must be conducted, with a view to producing recommendations for a unifying microservice component model capable of operating on different BDaaS platforms. Research, informed by accepted design principles such as reusability, loose coupling, cohesion, abstraction, composition, autonomy, portability and

distribution, needs to be conducted to advance towards a unifying set of models, methods and processes for big data processing in the cloud.

This research endeavours to fill this gap by producing a systematic and unified approach to developing Big Data as a Service (BDaaS) based on a microservices architecture. To achieve this aim, existing knowledge in the area of big data processing, microservices and architectures will be thoroughly reviewed. Informed by these findings, a unique classification for the different types of processing architectures will be proposed, and a new unifying microservices architecture for big data processing will be created. Additionally, a visual modelling notation and a formal specification language will also be produced to support the analysis and design of big data processing microservices.

SCOPE

This research will focus on big data processing, defined as the transformations which takes place after the data has been collected and before it is analysed (see fig. 1.). Storage, which can happen after collection, after processing, after analysing, or at any combination of these stages, is excluded from the scope of this project.

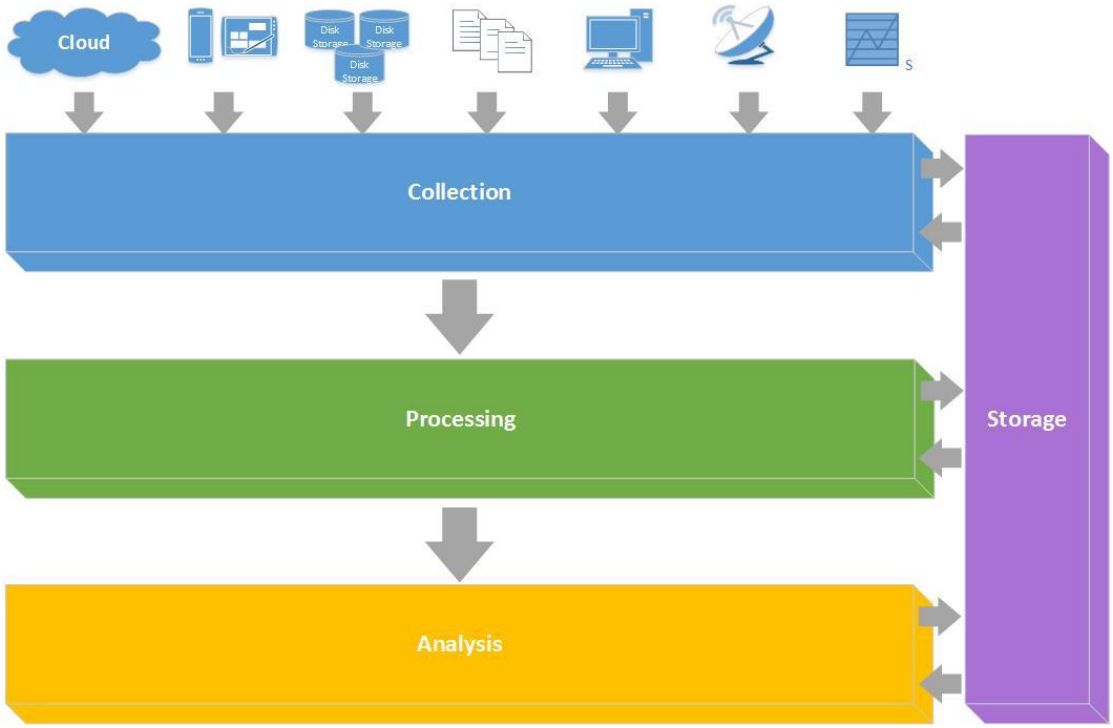

Figure 1. Big Data Architectural Layers

RATIONALE, AIM AND OBJECTIVES

RATIONALE

The diverse architectures and variety of technologies used to process big data make the learning curve for a developer working with big data high. A survey on IIOT adoption commissioned by the Industrial Internet Consortium reveals that 64% of senior IT executives believe integrating data from disparate sources and formats is amongst the biggest challenges of the Industrial IOT, and 36% of organisations have concerns around accessing the right skills and expertise (Soley et al., 2016). This shortage of highly skilled big data professionals is a worldwide problem which higher education providers are only beginning to address, according to research conducted at the Østfold University College (Due et al., 2015).

As indicated by Miller (2014), the demand for data engineers who are capable of integrating big data from different data sources is on the increase. However, since the types of data sources and technologies involved in big data processing are not standardised, it becomes challenging for universities to design a curriculum around the specific skills that the industry requires (Miller, 2014).

Calls have been made for a unifying development platform which could cope with the different challenges posed by big data, as well as a de facto programming language capable of expressing the different processing demands of big data analysis (Gorton, 2008). This solution would need to be capable of processing batch as well as stream data in order to reconcile the major big data processing models, and would represent an important contribution towards focusing the training and reducing the learning curve for IT professionals entering the field. It would also promote collaboration and reuse amongst developers working on a variety of microservices platforms.

RESEARCH QUESTIONS

This research seeks to investigate the major architectures designed for the processing of BDaaS, and to advance towards a unifying set of models, methods and processes for big data processing in the cloud. The primary questions of this research are thus:

- Does existing technology support the creation of unifying and reusable microservices capable of processing different types of big data from a variety of different sources?
- Which models, methods and processes need to be in place to support the design and development of these microservices?
- Would the proposed products of this research be useful to IT professionals in a real-world organisation?

AIM

The aim of this research is to produce a systematic and unified approach to developing Big Data processing pipelines in the cloud, based on reusable and scalable microservices.

OBJECTIVES

In order to achieve the aim of producing a systematic and unified approach to developing Big Data processing pipelines in the cloud, based on reusable and scalable microservices, I intend:

- to review and critically evaluate existing literature and approaches in the area of big data processing, microservices and architectures;
- to identify and evaluate key design principles for the development of a new modelling notation and microservices architecture for big data processing in the cloud, such as reusability, loose coupling, cohesion, abstraction, composition, autonomy, portability and distribution;
- to identify design patterns in existing big data frameworks and determine those applicable for a cloud-based microservices architecture for big data.
- to identify various behaviours of big data and to devise a new classification for existing big data processing patterns;
- to create a microservices architecture for the processing of big data by unified, reusable and scalable microservices;
- to evaluate the suitability of existing modelling languages such as UML/SoaML/SysML for the abstract representation, specification and verification of the proposed microservices for big data processing.
- to propose a visual modelling notation to represent cloud-based big data processing microservices;

- to identify metrics for cloud-based big data processing microservices and to evaluate them against these metrics;
- to develop a demonstrable prototype to evaluate the modelling notation and unified microservices architecture for big data processing in the cloud with a real-world case study (Leeds Beckett University, possibly Google Research, USA, and the Microsoft Azure Development Team).

The review of existing literature will be used to gain knowledge of the main existing methods for big data processing, and of existing big data and microservices architectures. It will also help identify the key design principles for big data processing and the design patterns applicable to a unifying BDaaS microservices architecture. A new level of abstraction to represent big data and a new microservices architecture for the processing of BDaaS by unified and reusable microservice components will represent unique contributions to existing knowledge, and will be devised using the information gathered in the literature review.

The next step will be to propose a visual modelling notation and a formal specification language capable of expressing the fundamental programming abstractions involved in big data processing and supporting the design of big data processing microservices.

The experimental part of this research will involve the creation of a prototype to demonstrate the proposed microservices architecture for processing different types of big data. This prototype will be evaluated technically, via benchmarks, and empirically, through a case study followed by qualitative data analysis.

RELATED WORK AND DEFINITIONS

RELATED WORK

A model for integrating BDaaS across different providers, called Neo-Metropolis, was proposed by H. Chen et al. (2016). This model is based on a kernel, which provides the platform's basic functionality, a periphery, composed of various service providers hosted on different clouds, and an edge, representing customers who utilise services and provide requirements (Chen, H. et al., 2016). Whilst the kernel would be fairly stable and backwards compatible, with stable releases, the periphery would be in constant development, or perpetual beta, and would be based on open-source code (H. Chen et al., 2016).

The Neo-Metropolis study conducted a case study based on Cisco's Intercloud Services Platform, which was, at the time of writing, the only commercial Neo-Metropolis implementation (H. Chen et al., 2016). Cisco Intercloud Services Platform provides a common interface for managing data and provisioning services across different cloud providers. Thus, the processing logic contained in services deployed to this platform can be reused, as data from different clouds can be plugged in as required (see fig. 2.) ("Cisco Intercloud Fabric: Hybrid Cloud with Choice, Consistency, Control and Compliance", 2016).

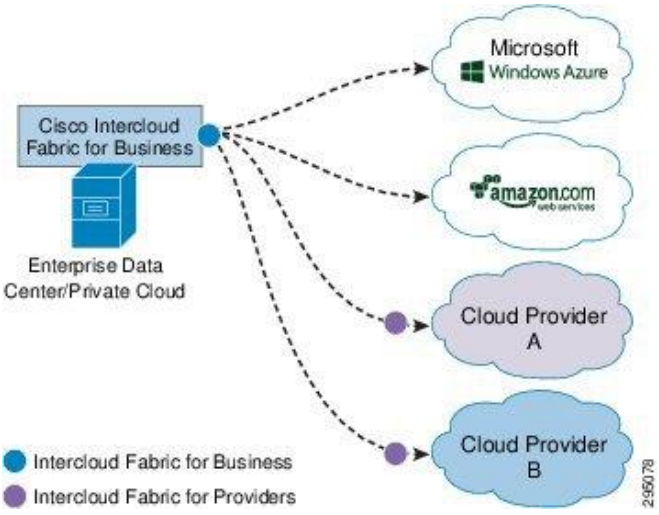

Figure 2. Service Provider Managed Hybrid Cloud ("Cisco Intercloud Fabric: Hybrid Cloud with Choice, Consistency, Control and Compliance", 2016)

This research complements the Neo-Metropolis solution, but differs from it in that it proposes an additional layer of abstraction to enable the processing of diverse types of

big data using a unified microservice component. Let us consider, for instance, a simple service that filters data based on some regular expression. A microservice developed within the Cisco Intercloud platform, built using Cloudera, Hortonworks, or another Hadoop implementation available within the platform, would run against Amazon EMR, Azure's HDInsight, or other Hadoop-based services hosted on different clouds. It would not, however, run against streaming data. If we wanted to reuse the same filtering logic to process stream data, the service would need to be re-written as a Storm or Spark-based service. This research proposes an abstraction for representing different types of big data which would enable a greater spectrum of reuse for big data processing microservices (see fig. 3.).

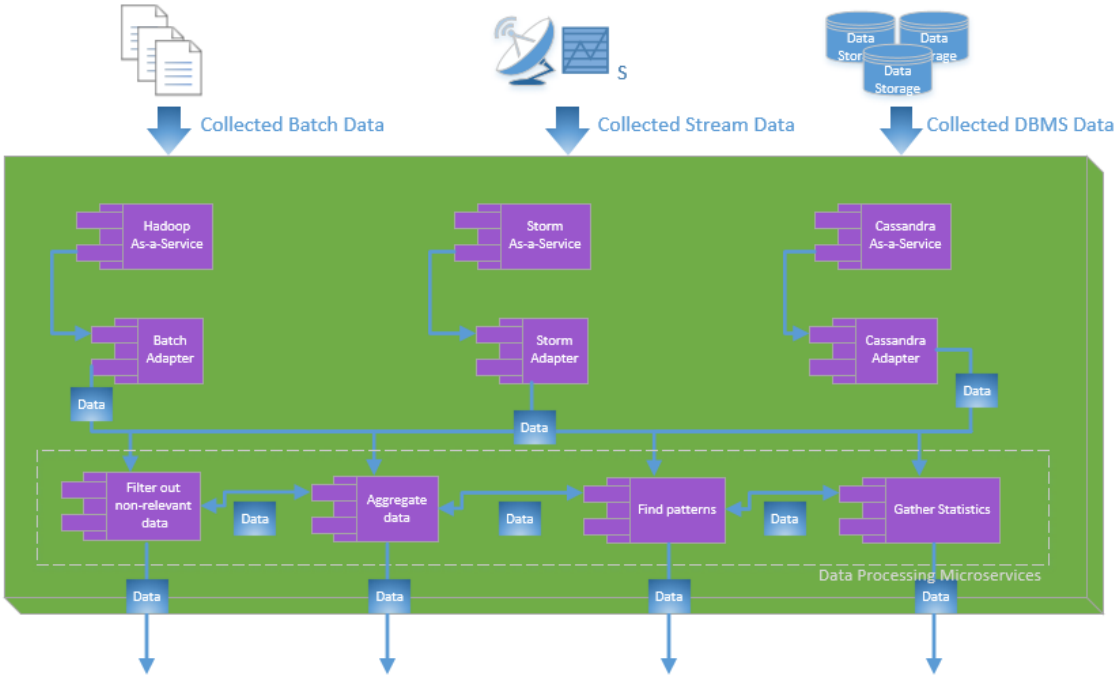

Figure 3. Proposed BDaaS Processing Layer

IBM's big data platform is a different type of solution to the Neo-Metropolis model. It is not a microservices type of architecture and does not provide inter-cloud integration, as did the Neo-Metropolis model, but it provides answers to some of the issues we identified previously, namely the interoperability of processing components across different types of big data. BigInsights, a Hadoop-based system consisting of a standard Hadoop kernel, plus a number of components built around it (Zikopoulos et at., 2012, p. 51), is a central part of IBM's big data platform. BigInsights integrates with Stream, a system designed to process real-time data, by means of adapters (Zikopoulos et at., 2012, p. 109).

The adapter used by IBM's big data platform to allow batch data to be processed as streams is particularly relevant to this research, as it uses the concept of windows of data to perform the conversion (Zikopoulos et at., 2012, p. 128). This concept is also present in the Java Stream API, for example, which uses windows based on element count to make an infinite stream finite ("Java Platform SE 8", 2014). Additionally, sliding time windows have been identified as a recurrent pattern in stream processing (Khare et al., 2015) and it has been suggested that a stream architecture enhanced with ways to reason about time could supersede the Lambda Architecture (Akidau, 2015). Previous research thus suggests that a common processing language for batch and stream data would need to incorporate the concept of windows of data.

This research aims to explore the use of the Adapter design pattern (Gamma et al., 1994) in a similar way to that implemented as part of IBM's big data platform. However, instead of translating from the batch layer to the stream layer and using the stream layer to process the data, or vice-versa, it proposes an additional layer of abstraction where the processing takes place (see fig. 3.) so as to facilitate the development, deployment and reuse of small processing units as microservices.

Google Cloud Dataflow is a managed cloud-based service which offers a unifying programming language for the processing of batch and stream data (Schmidt, 2015). It incorporates Millwheel, a stream-based framework built around the concepts of low watermarks and timers to extract accurate data from near-real-time streams (Akidau et al., 2013). Millwheel uses an abstraction called PCollection to represent bounded or unbounded stream or batch data ("PCollection" 2016). Unbounded data, i.e. very large batch data or infinite streams of data, is processed in stages through the use of windowing (see fig. 4) (Schmidt, 2015).

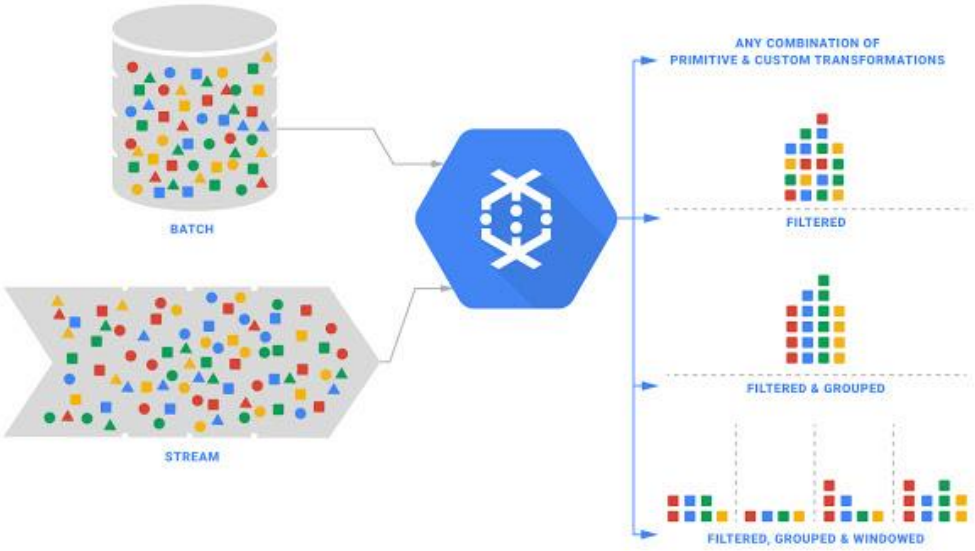

Figure 4. Cloud Dataflow provides a unified computation model for batch and streaming processing (Schmidt, 2015)

The Google Cloud Dataflow solution addresses the question of processing stream and batch data using a unifying programming language. The solution is proprietary, but Google has made the Cloud Dataflow SDK open source, and maintains it is possible to utilise it to build services which could be deployed on other clouds (Vambenepe, 2015). Cloudera has initiated a project to incorporate the Google Cloud Dataflow SDK into Apache Spark, but their project is still in its infancy and, at the time of writing this proposal, no substantial results have been found in the literature (Wills, 2015). This research shall conduct a rigorous academic assessment of these recent technological advancements, with a view to possibly utilising the Google Cloud Dataflow SDK as a programming language to support the proposed unifying microservices architecture for BDaaS processing. Furthermore, we shall contribute to existing research and practice by proposing a standardised set of models, methods and processes for the development of BDaaS processing microservices.

Other less closely related work, such as architectures designed to process batch or stream data, the lambda architecture and hybrid architectures, is detailed in the definitions section.

DEFINITIONS

The following terms will be used throughout this research as defined below.

ARCHITECTURE

As stated by (Bass et al., 2012, p. 26), software architecture is a new discipline and there are many existing definitions for it. Taylor et al. define it as

> *"the set of principal design decisions made during its development and any subsequent evolution." (Taylor et al., 2009, p. 1)*

This is a broad definition, which does not convey the idea of architecture as the representation of the structure of a software system. This research shall utilise the definition provided by Bass et al.

> *"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of these components, and the relationships among them." (Bass et al., 2012, p. 23)*

where a component would represent a microservice.

MICROSERVICES

Microservices are small deployable units which encapsulate functionality that can be used by other systems. A microservices architecture is usually defined as the opposite of a monolithic application architecture (Nadareishvili et al., 2016, p. 17), (Newman, 2015), (Bonér, 2016), i.e. an architecture guided by the principle of breaking down complex functionality into small independent units which collectively constitute a computing system. Sam Newman defines microservices as a collection of "small, autonomous services that work together" (2015, p. 16).

Microservices are cohesive, loosely coupled and composable (Miller, 2015) (Newman, 2015), making the architecture more resilient in the event of failure. They are also more agile, as their development and deployment takes comparatively less time than if the same functionality were to be implemented as part of a monolithic application (Miller, 2015). Since the components are cohesive, isolated and loosely coupled, there are fewer side-effects to implementing new code. Individual units can be replaced independently, so there is no need to bring a whole application down for upgrades or maintenance

(Miller, 2015). Given the distributed nature of big data systems, a microservices architecture is more flexible and resilient, and therefore more suitable than the traditional monolithic approach.

BIG DATA

Big data can be defined as data which somehow challenges the processing capabilities of current technology. These challenges are usually categorised around the three Vs: volume, velocity and variety (Casado & Younas, 2015), (Assunção et al., 2015), (Zikopoulos et al., 2013, p. 9) and others, with the occasional mention of additional Vs such as veracity, value and viability (Assunção et al., 2015).

The volume of data which lead internet companies such as Facebook and Netflix have accumulated has reached hundreds of petabytes (Krishnan & Tse, 2013), (Bronson et al., 2015), and it has been estimated that the largest big data company in the world, Google, holds over 10 exabytes of data (Lederman, 2016). This data is kept on disk, stored in data centres all over the world, posing significant architectural challenges when it comes to processing it in order to extract valuable information with an acceptable level of latency.

The velocity at which data is generated is also a significant factor when it comes to engineering applications which will consume and process this data. Netflix's data pipeline, for example, receives approximately 500 billion events a day, which amounts to 1.3 petabytes of incoming data that needs to be processed each day, in real time ("Evolution of the Netflix Data Pipeline," 2016). Facebook processes hundreds of gigabytes per second across hundreds of real-time data pipelines (G. Chen et al., 2016). These companies have invested in architectures which can consume incoming data at high velocity.

The most accepted classification for big data variety separates it into structured (usually stored in relational databases), semi-structured (data stored in NoSQL databases) and unstructured (Mohammed et al., 2016), (Casado & Younas, 2015). Assunção et al. add a mixed category to this classification (Assunção et al., 2015). Data which originates from surveillance cameras, social networks or tracking devices, for example, is diverse in structure from data stored in NoSQL databases, which is again diverse from data stored in relational databases. An architecture designed to cater for all types of big data

needs to take the variety factor into consideration, as one can no longer assume that data will be stored in a single relational database.

This research uses the term big data to refer to data characterised by the three Vs defined above.

BIG DATA PROCESSING PARADIGMS

Thomas Kuhn (1964) introduced the notion of scientific paradigms as

> *"…universally recognized scientific achievements that for a time provide model problems and solutions to a community of practitioners." (1964, viii)*

A paradigm as defined by Kuhn implies a level of consensus within the scientific community around concepts, practises and experiments over a period of time (1964).

In the field of big data processing, Casado & Younas (2015) identify three paradigms to explain the evolution of big data architectures: batch processing, stream processing, and the Lambda Architecture, which comprises batch plus stream processing.

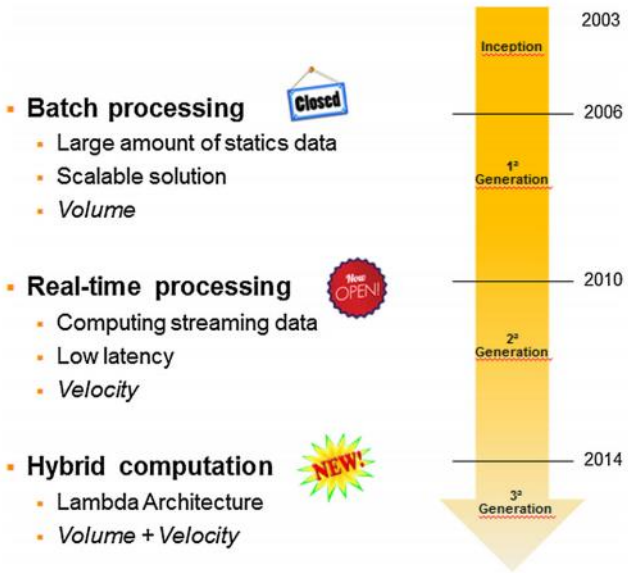

Figure 5. Processing paradigms (Casado and Younas, 2015)

At the time of writing their paper, the authors believed they were in the middle of the real-time or stream processing paradigm (see fig. 5.) (Casado & Younas, 2015). This is

explained through the number of real-time architectures that were being developed at the time. The Lambda Architecture and other hybrid computation models represented the future (Casado & Younas, 2015). This research argues that only batch processing characterises a paradigm in Kuhn's terms, as it was only during the batch period that there was enough agreement around concepts and processes to allow the scientific community to conduct normal science.

Contrary to what was suggested by Casado and Younas (2015), it is hard to trace a clear-cut line of when the stream processing paradigm ends and the hybrid computation paradigm begins. Twitter's Summingbird (Hausenblas, 2014) and Yahoo's Storm-Yarn (Evans & Feng, 2013), for example, are both early instances of the Lambda Architecture, whereas Apache Storm, one of the leading stream-based data processing frameworks, is still widely used and underwent a major release as late as 2016 (Goetz, 2016).

This research views the period after the end of the batch processing paradigm as one of crisis and emergence of new scientific theories, rather than of one of normal science, when a paradigm is able to provide adequate answers for the questions it defines (Kuhn, 1964). Through an evaluation of the emerging architectures and proposal of an original classification, this research aims to enhance current understanding of a period of scientific revolution in the field of big data processing.

BATCH PROCESSING

Batch processing was the first and is the most solidly established approach. It is based on the MapReduce algorithm and was designed at Google (Dean & Ghemawat, 2008), before going open-source as the Hadoop software framework. Hadoop is a distributed system for processing large volumes of data which is easy to use and extremely powerful (Dean & Ghemawat, 2008). It abstracts the complexities of parallelisation and inter-machine communication away from the user, who only needs to specify the map and reduce functions (Dean & Ghemawat, 2008). Hadoop can handle terabytes of data (O'Malley, 2008), but one of its main criticisms is its high latency and high start-up overhead (Stewart & Singer, 2012), which renders it ineffective for real-time systems.

Most cloud service providers offer some type of Hadoop managed service such as, for example, Amazon's Elastic MapReduce ("What is Amazon EMR?," 2016), Azure's HDInsight (Gronlund, 2016), or IBM's BigInsights (Zikopoulos et at., 2012, pp. 85-123).

Since this research aims to investigate big data processing in the context of a microservices architecture, it will focus on Hadoop provided as a service.

STREAM PROCESSING

Stream processing architectures evolved from the need to process real-time data. While batch processing is related to the volume of big data, stream processing relates to velocity. Real-time information such as which topics are trending on Twitter needs to come from data which is constantly being updated, with minimal latency. The notion of a data stream is an abstraction used to convey the nature of the data source: continuous and potentially infinite, and the way in which it is processed: in real-time (or close to real-time), before it is persisted to storage (Zikopoulos et al., 2013, p. 47).

Many reactive architectures were designed to cater for big data streams, such as Yahoo's S4 (Neumeyer et al., 2010), Apache Storm (Marz, 2014), Borealis (Abadi et al., 2005), StreamCloud (Gulisano et al., 2010), Stormy (Loesing et al., 2012), TelegraphCQ (Chandrasekaran et al., 2003) and Cyclops-React (McClean, 2015). Unlike the batch processing architectures, which are mostly centred around MapReduce, when it comes to stream-based architectures, there is no prevailing technology, although Apache Storm is a strong candidate (Casado & Younas, 2015). The main criticism when it comes to real-time stream-based systems is that they tend to compromise veracity or precision for velocity or ultra-low latency (Akidau, 2015).

Stream processing is offered as a service by cloud providers such as Amazon ("Amazon Kinesis Streams" 2016), Microsoft Azure ("Stream Analytics – Real-time data analytics," 2016) and IBM (Zikopoulos et al., 2013, pp. 127-162). This research shall concentrate on stream processing provisioned as a service.

LAMBDA ARCHITECTURE

The Lambda Architecture was presented by Marz & Warren to describe systems which would adopt a "best of both" approach to the batch/stream dichotomy. These architectures would use the stream layer for real-time data and the batch layer for historical data. The data would then be merged at query level (2015). Examples of this type of architecture can be found in Twitter's SummingBird (Hausenblas, 2014), Yahoo's Storm-Yarn (Evans & Feng, 2013), IBM's Big Data platform (Zikopoulos et al., 2013), Lambdoop (Tejedor, 2013), AllJoynLambda (Villari et al., 2014) and Facebook's

integration between its Puma, Swift and Stylus stream systems and its data stores (G. Chen et al., 2016). The main criticism to this approach is having to maintain two different complex architectures and having duplicate code in the two different layers (Kreps, 2014).

Examples of the lambda architecture provided as a service are Google Cloud Dataproc ("What is Google Cloud Dataproc?," 2016) and Azure's HDInsight Plus, which adds Apache Spark and Apache Storm clusters to the standard HDInsight Hadoop components (Shah, 2016). As with the batch and stream architectures, this research shall focus on the lambda architecture provided as a service.

HYBRID ARCHITECTURES

This research defines hybrid architectures as those which are batch-based but have been adapted to consume data from streams or those which are stream-based but have been adapted to consume historical data from batches. Examples of the first type are (Grinev et al., 2011), who modified the reduce function in MapReduce to accept push values, (Condie et al., 2010), who modified the map function to pipeline data to reducers as it is produced and (Brito et al., 2011), who reimplemented stateless stream processing operators as mappers and stateful operators as reducers. Examples of architectures which are stream-based, but capable of processing batch data with very low latency are Google's MillWheel (Akidau et al., 2013), elucidated in the Related Work section, and the combination of Kafka and Samza used at LinkedIn, which is built on the concepts of ordered data and replayable streams (Kreps, 2014).

This research shall concentrate on hybrid big data architectures provided as a service such as, for example, MillWheel, which is part of the Google Cloud Dataflow platform.

DATABASE-BASED ARCHITECTURES

Database-based architectures are built around databases or data stores and may or may not be adapted to consume stream data. Examples of these architectures are Percolator, a system designed to incrementally update Google's web search indexes (Peng & Dabek, 2010), SnappyData, which combines Apache Storm with GemFire, a transactional store which keeps data in RAM memory across distributed nodes for fast access (Ramnarayan et al., 2016), and (Simmonds et al., 2014) which uses Cassandra

distributed across multiple nodes in the cloud and minimises latency through the use of efficient indexing patterns.

Database-based architectures generally underperform when compared to the other types of architecture described previously (Peng and Dabek, 2010), so they are not offered as cloud-based services by the main commercial providers. The evaluation of (Simmonds et al., 2014)'s research was performed using the Amazon cloud, but they did not offer a comparison with other types of processing architectures. Although database-based big data processing architectures are not typically offered as a service, this research proposes to examine them in order to acquire a thorough understanding of the subject matter, and to gather knowledge to create a new classification for big data processing architectures.

METHODOLOGY

Briony Oates identifies six strategies for carrying out research: survey, design and creation, experiment, case study, action research and ethnography (Oates, 2005). This research adopts the design and creation strategy, and uses a case study for its evaluation.

PHILOSOPHICAL APPROACH

Approach can be defined as the researcher's perspective, or the manner in which they conduct their research (Galliers, 1985, p. 147). Generally, the literature identifies two competing approaches to research: positivist/empirical and interpretivist/constructivist. Positivists focus their research on scientific experiments, characterised by repeatability, reductionism and refutability. Interpretivists, on the other hand, recognise that reality is formed by individual and social constructs, and believe there can be many possible interpretations for the same phenomena (Oates, 2005, pp. 281-296), (Leedy & Ormrod, 2000), (Galliers, 1985).

Although the positivist approach still prevails in the realms of the natural and engineering sciences, as well as computer science and software engineering (Lee et al., 2014), it is recognised that positivism has been generally ineffective in social sciences (Hirschheim, 1985). By taking the viewpoint that computer systems are social rather than technical systems (Land, 1992), (Hirschheim, 1985), this research shall reject positivism in favour of one of its variants: post-positivism, along the lines of Karl Popper (Popper, 1968).

Popper shared many of the tenets of positivism such as repeatability and reliance on rigorous experiments, but he believed there was a fundamental problem with the validity of statements gathered from experience. The inductive process could not, according to him, lead to universal truths about the world. It could only lead to imperfect knowledge which will one day be superseded. A meaningful scientific hypothesis, according to Popper, is one which is falsifiable, that is to say one which could be invalidated by experience. If a given hypothesis cannot be invalidated by any possible experiment, then it does not enhance the knowledge we have of the world – it is not scientific. From this perspective, we can derive that what is possible to know about the world is not that something is true (true or false are not empirical concepts for Popper), but that, so far, experiments have not proven it to be false (Popper, 1968). The current research shall be

undertaken under this philosophical stance. Care will be taken to ensure that the hypotheses formulated in the development and evaluation of the research products are empirically falsifiable. Additionally, the theories, models and classifications to emerge from this study will be understood, not as universal truths, but as improvements to existing theories.

This research agrees with Roth and Mehta (Roth & Mehta, 2002) in that there are benefits to be reaped from combining different philosophical perspectives in the evaluation of a research product. In particular, the evaluation and discussion of the results of the case study proposed shall benefit from an interpretivist perspective, as the outcomes observed will be influenced by a number of different variables which cannot be as easily controlled as in a laboratory experiment. This research recognises the value of considering different interpretations for qualitative data, and ascribes significance to the impact that a scientist may have in the case being studied.

PROTOTYPE DEVELOPMENT

The development methodology used in the creation of a new microservices architecture for big data processing in the cloud, as well as in the case study implementation will be prototype-based. There will be an initial analysis and design phase in which the knowledge gathered from the literature will be processed and design ideas put forward for examination. This shall be followed by the creation of a prototype to demonstrate the main concepts of the proposed solution.

The prototype will be further enhanced through additional cycles of analysis, design, implementation and testing. The choice of a prototype-based methodology instead of, for example, a waterfall-based approach, was taken to allow for early experimentation with different solutions. This warrants for better distribution of the time available and mitigates the risk of spending excessive time on design and analysis, to the detriment of implementation and testing (Oates, 2005, p. 114).

A pure agile approach to development was considered but, due to the importance of design and analysis to this research, it was deemed unsuitable. While the agile approach focuses on delivery and measures progress by the software that gets delivered (Beck et al., 2001), this research shall focus on design and analysis and how these can be enhanced with each iterative development cycle.

REQUIREMENTS

The development of the proposed microservices architecture for big data processing requires:

- a source of big batch data;
- a source of fast streaming data;
- a clustered Hadoop environment available as a service;
- a Stream environment available as a service;
- a development environment with 4-8 virtual machines, preferably hosted in different clouds;
- a minimum of 16 cores of CPU and 1TB of storage.

An application has been submitted to the Azure for Research program for cloud resources meeting the above specifications.

An application will be submitted to the Open Science Data Cloud (OSDC), a scientific community which provides support for researchers, for:

- General Compute Resources, which include 1TB of storage and 16 cores of CPU;
- Access to the Public Data Commons, which contains 1PB of public data in a variety of disciplines;
- Hadoop Resources.

Alternative providers of cloud resources have been considered: Amazon AWS and Google Cloud Platform, but preference will be given to Azure for Research and the Open Science Data Cloud as they are the only services which are free of charge for researchers.

EVALUATION

The prototype proposed as part of this research will be evaluated empirically by means of benchmarks and a case study.

BENCHMARKS

The microservices architecture for big data processing proposed in this research will be evaluated through benchmarks and compared to industry standard solutions for big data processing.

A minimum of three fundamental quality metrics will be identified and used to set up the benchmarks, reflecting key software architecture principles such as reusability, loose coupling, cohesion, abstraction, composition, autonomy, portability and distribution.

The proposed microservices architecture shall be compared to that of Hadoop provided as a service, as well as the Google Cloud Dataflow.

CASE STUDY

This research proposes to evaluate the new microservices architecture, the visual modelling notation and the formal specification language created by means of a case study, which will also serve as triangulation for the benchmark evaluation.

Although there have been increasing demands for real-world evaluation of computer systems (Oates, 2005, p. 111), this is not undertaken very often in computer science. In a survey of 400 research papers, (Tichy et al., 1995) concluded that computer scientists published significantly less empirically validated results when compared to optical engineers and neural scientists. They regard this as a limitation which needs to be corrected if computer science is to enjoy the same level of recognition as engineering and the natural sciences (Tichy et al., 1995). A broader survey was conducted more recently where 1500 scientific papers in the areas of computer science, software engineering and information systems were analysed. The conclusion was similar: computer science and software engineering tend to focus on creating new products, but there is very little effort expended on evaluation (Glass et al., 2009). In light of these shortcomings in the field, this research aims to provide real-world evaluation of its products by conducting a case study.

The case study proposed shall allow us to evaluate:

- the effectiveness of the microservices architecture in processing big data in a real- world setting;
- the usability of the microservices architecture to software developers;
- the expressiveness of the visual modelling notation and specification language.

Furthermore, the case study will be used to gauge the relevance of this research to practitioners, as advocated by (Rosemann & Vessey, 2008) in their repudiation of the rigour versus relevance dichotomy. Although they suggest focus groups as the ideal means of conducting applicability checks, due to the importance of leaving the research object unchanged and the fact that a focus group doesn't need a research lifecycle of its own (Rosemann & Vessey, 2008), this research believes a small scale case study meets the required criteria while painting a richer picture of the object being studied.

CASE DESCRIPTION

A case study will be conducted within the Estates Services department at Leeds Beckett University. As part of the case study, a prototype of the proposed microservices architecture shall be implemented for a limited set of data with the typical volume, velocity and variety characteristics.

The case study will then be described and the microservices architecture, visual modelling notation and specification language presented to the target population through appropriate documentation. This shall be followed by semi-structured interviews with the target population.

EVALUATION CRITERIA

This case study will be conducted in order to verify whether:

- The microservices architecture can process stored data with an acceptable level of latency in a real-world scenario.
- The microservices architecture can process streaming data with an acceptable level of accuracy in a real-world scenario.
- The microservices are decoupled from their underlying platforms in a real-world scenario.
- The visual modelling notation and specification language are accessible, expressive and useful to business analysts, project managers and developers.
- The microservices architecture is accessible to developers who have limited experience with big data analytics.

SELECTION CRITERIA

This case study has been selected for being critical, meaning it has the particular conditions that allow the testing of the research's hypotheses (Dubé & Paré, 2003). The Estates Services department at Leeds Beckett University manages data which has the fundamental characteristics of big data: volume (log files with operational data that spans several years), velocity (data from meters, that is sampled and streamed in real-time) and variety (device log exports, for example, contain unstructured data, as the format is specific to each manufacturer).

A needs analysis exercise was conducted with the Engineering Systems Manager at Leeds Beckett University to gather preliminary information for this case study. It revealed the existence of a study conducted in 2012 to assess the energy efficiency of one of the university's data centres by calculating its Power Usage Effectiveness (PUE) (Pattinson, 2012). The real-time sampling rate used for this study was, however, limited by the technology available at the time. It is possible that, by modifying the current sampling approach to one of higher volume and velocity, the calculation will become more

accurate and therefore suitable to other controlled environments such as laboratories or large colocation data centres.

It should be stated that convenience has also influenced the selection of this case study. This research is being conducted at Leeds Beckett University, where the researcher is a student and also an employee. Whereas this could result in a stronger bias when selecting the target population, conducting the interviews and analysing the results, local knowledge of the case could be seen as a positive factor, as the researcher has easy access to the people and resources involved and could take the stance of a participant observer (Thomas, 2013).

In order to mitigate researcher bias during data collection, this research shall present the data collected to the target population for feedback and checking. This method has been identified by Galliers as essential in order to reduce misinterpretation and identify researcher bias (Galliers, 1985).

SAMPLING

The sampling strategy for this case study will be purposive sampling, as it is important that participants selected have the right level of expertise. It is acknowledged that the selection will be influenced by the researcher's own judgement. However, the advantages of being able to obtain a sample which is representative of specific viewpoints within the organisation outweigh the disadvantage of having greater researcher bias than if probability sampling methods had been utilised.

The proposed sample for this case study shall comprise:

- business analysts;
- head of sustainability;
- engineering systems manager and officers;
- software developers.

DATA COLLECTION METHODS

Case studies should strive to employ a variety of data collection methods (Dubé & Paré, 2003), (Kaplan & Maxwell, 2005), (Dawson, 2009). This research proposes to collect data by using semi-structured interviews, observations and documents. Interviews will

be recorded and transcribed, observations will be recorded as field notes and documents will be referenced in the bibliography.

RESEARCH ETHICS

As the case study proposed involves human participants, Research Ethics Approval will be needed. A risk checklist has been completed, which provisionally classified this research as Risk Category 2. An application for Research Ethics Approval shall be prepared and submitted to the Local Research Ethics Co-ordinator a year before the case study is due to commence.

DATA ANALYSIS

The data collected in this case study will be stored and analysed with the help of the NVivo qualitative data analysis software. Although there can be disadvantages associated with using specialised software to manage research data, such as having to spend time learning the software or adding a layer of separation between the researcher and the data (Creswell, 2012), this research believes these are outweighed by the benefits of being able to organise how data is stored, using version control and performing searches for expressions and patterns.

On a first pass, the data will be read and classified according to the themes proposed by Oates:

- segments that bear no relation to the overall research purpose;
- segments that provide information describing the research context;
- segments that are relevant to the research question (Oates, 2005, p. 268).

The analysis shall focus on the third theme, as suggested by Oates, and further categorisation will be carried out inductively to identify, refine and connect emerging themes (Oates, 2005, 268-270). This shall be done through progressive cycles of data organisation, reflexion, categorisation, interpretation and presentation, as advocated by Creswell (2012, pp. 150-154).

Analysis of the second theme will also be carried out, as suggested by Creswell (2012, 150-154), in order to provide a rich picture of the case, its settings and participants.

ADDITIONAL CASE STUDY

If time permits, an additional case study will be conducted with Microsoft Azure, who are sponsors of this project. Their cloud development team will be contacted and asked to evaluate the products of this research.

RISKS

A risk analysis exercise was conducted using the format proposed by Sommerville (Sommerville, 2010, pp. 600-601). Table 1 shows the risks which could impact this project and the strategies identified for their management.

Risk	Probability	Effects	Strategy
The cloud resources applied for are not granted.	Moderate	Serious	Investigate the possibility of obtaining department funding for equivalent resources hosted on Amazon AWS, Azure or the Google Cloud Platform.
The time allocated to the development of the prototype is underestimated.	High	Tolerable	Reduce the scope of the prototype and present the design/plan of the solution, instead of a full implementation.
The time allocated to familiarisation with the technologies involved is underestimated.	High	Serious	Apply early for the cloud resources needed and start the familiarisation process a year before the development of the prototype is due. Attend the NVivo training course offered by Leeds Beckett University as part of their research programme.
The organisation selected is restructured	Low	Serious	Maintain communication with the organisation and, if there are any changes and the

and the case study is denied.			case study is denied, look for an alternative organisation. In the worst case scenario, use graduate students with the desired skill set as an alternative.
There are delays in conducting the interviews due to cancellations.	High	Tolerable	More buffer days were allocated to the year when the interviews will be conducted, when compared to other years. This was done in order to absorb delays and cancellations.

Table 1. Risk Analysis

The risk analysis will be updated every year for presentation to the progression panel.

STRATEGY

RESEARCH PLAN

Fig. 6 shows a summarised version of the PhD Gantt Chart for this project, and Fig. 7 shows the project's timeline. A complete project plan has been submitted as a separate document.

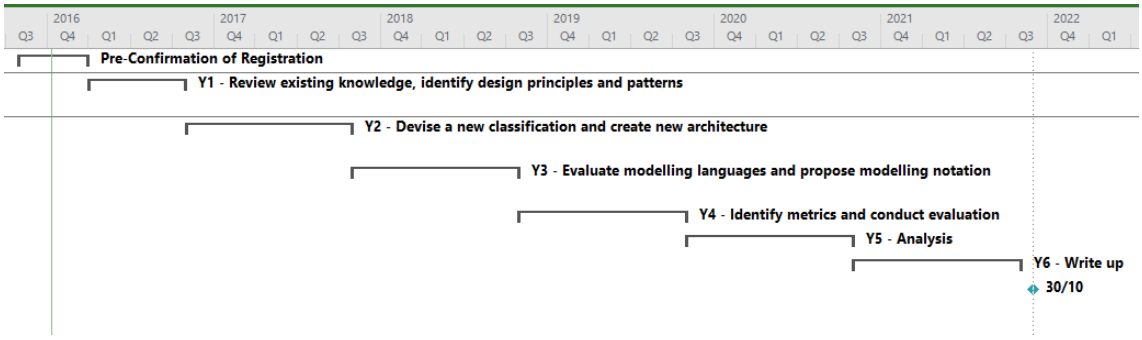

Figure 6. PhD Gantt Chart

The project plan will be kept up-to-date and shared with both supervisors. It will be presented to the progression panel at each progression meeting.

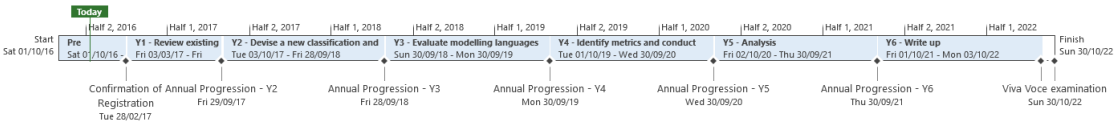

Figure 7. PhD Timeline

YEAR 1

The first year of this research will be dedicated to reviewing the existing knowledge, identifying and evaluating key design principles and establishing appropriate design patterns for BDaaS processing microservices.

The initial literature review will be conducted at this stage and the draft chapter will be written and presented for evaluation and feedback. However, as the literature review is an ongoing process, there are also 13 additional reading weeks distributed around the 6 years of this project, each followed by a literature review update week.

There are 28 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Pre-Confirmation of Registration	**151 days**	**Sat 01/10/16**	**Tue 28/02/17**
Define research topic	**21 days**	**Sat 01/10/16**	**Fri 21/10/16**
Literature search (confirm gap)	**15 days**	**Fri 21/10/16**	**Fri 04/11/16**
Prepare rationale, aims and objectives	**22 days**	**Tue 01/11/16**	**Tue 22/11/16**
Prepare strategy, methodology and evaluation	**12 days**	**Mon 21/11/16**	**Fri 02/12/16**
Apply for Research Ethics	7 days	Sat 03/12/16	Fri 09/12/16
Prepare literature review	**18 days**	**Tue 29/11/16**	**Fri 16/12/16**
Prepare research proposal	**22 days**	**Fri 20/01/17**	**Fri 10/02/17**
Submit research proposal	1 day	Mon 27/02/17	Mon 27/02/17
Confirmation of Registration	1 day	Tue 28/02/17	Tue 28/02/17
Research Training	**92 days**	**Wed 02/11/16**	**Wed 01/02/17**
Holidays	**23 days**	**Sun 25/12/16**	**Mon 16/01/17**
Buffers	**72 days**	**Sat 17/12/16**	**Sun 26/02/17**
Supervisory meeting	1 day	Mon 23/01/17	Mon 23/01/17
Y1 - Review existing knowledge, identify design principles and patterns	**211 days**	**Fri 03/03/17**	**Fri 29/09/17**
Review existing knowledge	**36 days**	**Fri 03/03/17**	**Fri 07/04/17**
Identify and evaluate key design principles for BDaaS processing microservices	**60 days**	**Tue 04/04/17**	**Fri 02/06/17**
Identify design patterns in existing big data frameworks and determine those applicable for a BDaaS microservices architecture	**102 days**	**Tue 30/05/17**	**Fri 08/09/17**
Reading weeks	**111 days**	**Tue 02/05/17**	**Sun 20/08/17**
Holidays	**23 days**	**Sat 15/07/17**	**Sun 06/08/17**
Prepare for Annual Progression	20 days	Sat 09/09/17	Thu 28/09/17
Annual Progression - Y2	1 day	Fri 29/09/17	Fri 29/09/17

Figure 8. Year 1

YEAR 2

The second year of this research will be dedicated to devising a new classification for big data processing patterns and creating a new microservices architecture for the processing of BDaaS.

The creation of a microservices architecture will include one cycle of initial planning, three prototype implementation cycles and a final cycle for documentation, each taking place over 18 days and including presentation of results, evaluation and feedback. An additional 49 days have been allocated to writing up the corresponding chapter, which also include presentation of results, followed by evaluation and feedback.

This year also includes the preparation of a paper for publication, based on the partial findings of this research. This will take place over 51 days, following the completion of the microservices architecture prototype.

There are 42 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Y2 - Devise a new classification and create new architecture	**361 days**	**Tue 03/10/17**	**Fri 28/09/18**
Apply for OSDC resources	28 days	Tue 03/10/17	Mon 30/10/17
Identify various behaviours of big data and to devise a new classification for existing big data processing patterns	**46 days**	**Tue 03/10/17**	**Fri 17/11/17**
Plan microservices architecture for BDaaS	**18 days**	**Tue 16/01/18**	**Fri 02/02/18**
Implement prototype to demonstrate the new microservices architecture for BDaaS	**46 days**	**Tue 30/01/18**	**Fri 16/03/18**
Write Documentation on new microservices architecture for BDaaS	**18 days**	**Tue 13/03/18**	**Fri 30/03/18**
Write chapter on new library extension	**49 days**	**Sat 14/04/18**	**Fri 01/06/18**
Write a paper for publication	**51 days**	**Tue 29/05/18**	**Wed 18/07/18**
Reading weeks	**268 days**	**Sat 18/11/17**	**Sun 12/08/18**
Holidays	**256 days**	**Sat 23/12/17**	**Tue 04/09/18**
Buffers	**287 days**	**Sat 02/12/17**	**Fri 14/09/18**
Prepare for annual progression	13 days	Sat 15/09/18	Thu 27/09/18
Annual Progression - Y3	1 day	Fri 28/09/18	Fri 28/09/18

Figure 9. Year 2

YEAR 3

The third year of this research will be dedicated to evaluating existing modelling languages and creating a new visual modelling notation for the abstract representation, specification and verification of the proposed microservices for big data processing.

There are 35 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Y3 - Evaluate modelling languages and propose modelling notation	**366 days**	**Sun 30/09/18**	**Mon 30/09/19**
Evaluate the suitability of existing modelling languages such as UML/SoaML/SysML	**48 days**	**Sun 30/09/18**	**Fri 16/11/18**
Propose a new modelling notation	**109 days**	**Tue 13/11/18**	**Fri 01/03/19**
Propose a formal specification language	**60 days**	**Tue 26/02/19**	**Fri 26/04/19**
Write Documentation on new modelling notation and specification language	**46 days**	**Tue 23/04/19**	**Fri 07/06/19**
Write chapters on new modelling notation and specification language	**47 days**	**Mon 29/07/19**	**Fri 13/09/19**
Reading Weeks	**144 days**	**Tue 29/01/19**	**Fri 21/06/19**
Holidays	**219 days**	**Sat 22/12/18**	**Sun 28/07/19**
Buffers	**217 days**	**Sat 01/12/18**	**Fri 05/07/19**
Buffer	21 days	Sat 01/12/18	Fri 21/12/18
Buffer	14 days	Sat 22/06/19	Fri 05/07/19
Prepare for Annual Progression	16 days	Sat 14/09/19	Sun 29/09/19
Annual Progression - Y4	1 day	Mon 30/09/19	Mon 30/09/19

Figure 10. Year 3

YEAR 4

The fourth year of this research is dedicated to identifying metrics based on software quality principles and evaluating the products of this research by means of benchmarks and a real-world case study.

There are 45 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Y4 - Identify metrics and conduct evaluation	**366 days?**	**Tue 01/10/19**	**Wed 30/09/20**
Identify metrics based on software quality principles	**18 days?**	**Tue 01/10/19**	**Fri 18/10/19**
Benchmark Evaluation	**158 days**	**Tue 15/10/19**	**Fri 20/03/20**
Set up benchmarks based on metrics	**39 days**	**Tue 15/10/19**	**Fri 22/11/19**
Analyse benchmarks	**18 days**	**Tue 19/11/19**	**Fri 06/12/19**
Write chapter on benchmark evaluation	**47 days**	**Mon 03/02/20**	**Fri 20/03/20**
Empirical evaluation	**149 days**	**Tue 17/03/20**	**Wed 12/08/20**
Review Research Ethics	7 days	Tue 17/03/20	Mon 23/03/20
Case study implementation	**46 days**	**Tue 17/03/20**	**Fri 01/05/20**
Analysis and design	**18 days**	**Tue 17/03/20**	**Fri 03/04/20**
Design, implementation and testing cycles	**32 days**	**Tue 31/03/20**	**Fri 01/05/20**
Data generation	**107 days**	**Tue 28/04/20**	**Wed 12/08/20**
Identify respondents	**18 days**	**Tue 28/04/20**	**Fri 15/05/20**
Prepare information pack for interviewees	**18 days**	**Tue 12/05/20**	**Fri 29/05/20**
Conduct interviews	**79 days**	**Tue 26/05/20**	**Wed 12/08/20**
Reading weeks	**152 days**	**Mon 20/01/20**	**Fri 19/06/20**
Holidays	**265 days**	**Tue 24/12/19**	**Sun 13/09/20**
Buffers	**259 days**	**Sat 07/12/19**	**Fri 21/08/20**
Prepare for annual Progression	16 days	Mon 14/09/20	Tue 29/09/20
Annual Progression - Y5	1 day	Wed 30/09/20	Wed 30/09/20

Figure 11. Year 4

YEAR 5

The fifth year of this research will be spent analysing the results of the case study, writing the relevant chapter and writing a chapter to discuss the results of the research.

This year includes the preparation of a second paper for publication, based on the findings of the case study. This will take place over 82 days, following the completion of the discussion of results.

There are 29 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Y5 - Analysis	**364 days**	**Fri 02/10/20**	**Thu 30/09/21**
Qualitative Data Analysis	**50 days**	**Fri 02/10/20**	**Fri 20/11/20**
Cycle 1	**22 days**	**Fri 02/10/20**	**Fri 23/10/20**
Reflect, Categorise and Interpret data using NVivo	17 days	Fri 02/10/20	Sun 18/10/20
Present data analysis	1 day	Mon 19/10/20	Mon 19/10/20
Feedback on data analysis	5 days	Mon 19/10/20	Fri 23/10/20
Cycle 2	**18 days**	**Tue 20/10/20**	**Fri 06/11/20**
Cycle 3	**18 days**	**Tue 03/11/20**	**Fri 20/11/20**
Write results of data analysis	**25 days**	**Tue 17/11/20**	**Fri 11/12/20**
Write chapter on empirical evaluation	**47 days**	**Mon 18/01/21**	**Fri 05/03/21**
Write chapter on discussion of results	**56 days**	**Sat 20/03/21**	**Fri 14/05/21**
Write a paper for publication	**82 days**	**Sat 15/05/21**	**Wed 04/08/21**
Reading weeks	**14 days**	**Sat 06/03/21**	**Fri 19/03/21**
Holidays	**270 days**	**Thu 24/12/20**	**Sun 19/09/21**
Buffers	**259 days**	**Sat 12/12/20**	**Fri 27/08/21**
Prepare for Annual Progression	10 days	Mon 20/09/21	Wed 29/09/21
Annual Progression - Y6	1 day	Thu 30/09/21	Thu 30/09/21

Figure 12. Year 5

YEAR 6

The sixth and final year of this research will be spent writing up the final thesis. Each chapter has been allocated between 35 and 64 days of preparation time, varying according to complexity, and will be submitted twice for evaluation and feedback. The chapters to be written during the final year are:

- Research Methodology
- Conclusion
- Limitations of Research
- Suggestions for Further Work
- Introduction
- Abstract, Keywords and References

There are 27 days allocated to preparation of the final thesis for submission and 27 buffer days to accommodate eventualities.

Task Name	Duration	Start	Finish
Y6 - Write up	**368 days**	**Fri 01/10/21**	**Mon 03/10/22**
Write research methodology chapter	**50 days**	**Fri 15/10/21**	**Fri 03/12/21**
Write conclusion	**47 days**	**Mon 10/01/22**	**Fri 25/02/22**
Write limitations of research	**35 days**	**Sat 26/02/22**	**Fri 01/04/22**
Write suggestions for further work	**35 days**	**Sat 02/04/22**	**Fri 06/05/22**
Write introduction	**49 days**	**Sat 07/05/22**	**Fri 24/06/22**
Write abstract, keywords and references	**64 days**	**Mon 01/08/22**	**Mon 03/10/22**
Reading weeks	**14 days**	**Fri 01/10/21**	**Thu 14/10/21**
Holidays	**227 days**	**Fri 17/12/21**	**Sun 31/07/22**
Buffers	**217 days**	**Sat 04/12/21**	**Fri 08/07/22**
Prepare thesis for submission	**27 days**	**Sat 03/09/22**	**Thu 29/09/22**
Submit thesis	1 day	Fri 30/09/22	Fri 30/09/22
Viva Voce examination	1 day	Sun 30/10/22	Sun 30/10/22

Figure 13. Year 6

BIBLIOGRAPHY

Abadi, D.J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., others, 2005. The Design of the Borealis Stream Processing Engine., in: CIDR. pp. 277–289.

Akidau, T., 2015. Have Your Cake and Eat It Too -- Further Dispelling the Myths of the Lambda Architecture [Online]. URL https://www.infoq.com/presentations/millwheel#downloadPdf (accessed 10.28.16).

Akidau, T., Whittle, S., Balikov, A., Bekiroğlu, K., Chernyak, S., Haberman, J., Lax, R., McVeety, S., Mills, D., Nordstrom, P., 2013. MillWheel: fault-tolerant stream processing at internet scale. Proc. VLDB Endow. 6, 1033–1044. doi:10.14778/2536222.2536229.

Amazon Kinesis Streams [Online], 2016. URL https://aws.amazon.com/kinesis/streams/ (accessed 11.26.16).

Assunção, M.D., Calheiros, R.N., Bianchi, S., Netto, M.A.S., Buyya, R., 2015. Big Data computing and clouds: Trends and future directions. J. Parallel Distrib. Comput. 79–80, 3–15. doi:10.1016/j.jpdc.2014.08.003.

Bass, L., Clements, P. & Kazman, R. (2012) Software Architecture in Practice. 3 edition. Upper Saddle River, NJ, Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mallor, S., Shwaber, K., Sutherland, J., 2001. The Agile Manifesto.

Bonér, J., 2016. Reactive Microservices Architecture. O'Reilly Media, Inc.

Brito, A., Martin, A., Knauth, T., Creutz, S., Becker, D., Weigert, S., Fetzer, C., 2011. Scalable and Low-Latency Data Processing with Stream MapReduce. 2011 IEEE Third Int. Conf. Cloud Comput. Technol. Sci. CloudCom 48.

Bronson, N., Lento, T., Wiener, J.L., 2015. Open data challenges at Facebook. IEEE, pp. 1516–1519. doi:10.1109/ICDE.2015.7113415.

Butzin, B., Golatowski, F. & Timmermann, D. (2016) Microservices approach for the internet of things. In: 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). pp.1–6.

Casado, R., Younas, M., 2015. Emerging Trends and Technologies in Big Data Processing. Concurr. Comput.: Pract. Exper. 27, 2078–2091. doi:10.1002/cpe.3398.

Chan, Y., Gray, I., Wellings, A., Audsley, N.C., 2014. Exploiting Multicore Architectures in Big Data Applications: The JUNIPER Approach. Program. Issues Heterog. Multicores MULTIPROG.

Chandrasekaran, S., Shah, M.A., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S.R., Reiss, F., 2003. TelegraphCQ: continuous dataflow processing. ACM Press, p. 668. doi:10.1145/872757.872857.

Chen, G.J., Yilmaz, S., Wiener, J.L., Iyer, S., Jaiswal, A., Lei, R., Simha, N., Wang, W., Wilfong, K., Williamson, T., 2016. Realtime Data Processing at Facebook. ACM Press, pp. 1087–1098. doi:10.1145/2882903.2904441.

Chen, H.M., Kazman, R., Haziyev, S., Kropov, V. & Chtchourov, D. (2016) Big Data as a Service: A Neo-Metropolis Model Approach for Innovation. In: 2016 49th Hawaii International Conference on System Sciences (HICSS). pp.5458–5467.

Cisco Intercloud Fabric: Hybrid Cloud with Choice, Consistency, Control and Compliance (2016) [Online]. Available from: <http://www.cisco.com/c/en/us/td/docs/solutions/Hybrid_Cloud/Intercloud/Intercloud_Fabric.pdf> [Accessed 25 November 2016].

Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Gerth, J., Talbot, J., Elmeleegy, K., Sears, R., 2010. Online aggregation and continuous query support in MapReduce. ACM Press, p. 1115. doi:10.1145/1807167.1807295.

Creswell, J.W., 2012. Qualitative Inquiry and Research Design: Choosing Among Five Approaches, 3rd Revised edition edition. ed. SAGE Publications, Inc, Los Angeles.

Dawson, D.C., 2009. Projects in Computing and Information Systems: A Student's Guide, 2 edition. ed. Addison Wesley, Harlow, England; New York.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 107. doi:10.1145/1327452.1327492.

Dubé, L. and Paré, G. (2003) Rigor in Information Systems Positivist Case Research: Current Practices, Trends, and Recommendations. MIS Q., 27 (4), pp.597–635.

Due, B., Kristiansen, M., Colomo-Palacios, R. & Hien, D.H.T. (2015) Introducing big data topics: a multicourse experience report from Norway. In: ACM Press, pp.565–569. Available from: <http://dl.acm.org/citation.cfm?doid=2808580.2808667> [Accessed 1 November 2016].

Evans, B., Feng, A., 2013. Storm-YARN Released as Open Source | YDN Blog - Yahoo [Online]. URL https://developer.yahoo.com/blogs/ydn/storm-yarn-released-open-source-143745133.html (accessed 10.28.16).

Evans, P. and Annunziata, M. (2012) Industrial Internet: Pushing the Boundaries of Minds and Machines.

Galliers, R. (1985) Choosing Information Systems Research Approaches. Research Methods in Information Systems, pp.144–162.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. & Booch, G. (1994) Design Patterns: Elements of Reusable Object-Oriented Software. 1 edition. Reading, Mass, Addison-Wesley Professional.

Glass, R.L., Vessey, I. and Ramesh, V. (2009) RESRES: The story behind the paper 'Research in software engineering: An analysis of the literature'. Information and Software Technology, 51 (1), pp.68–70.

Goetz, T., 2016. Storm 1.0.0 released [Online]. URL http://storm.apache.org/2016/04/12/storm100-released.html (accessed 11.6.16).

Gorton, I., 2008. Software Architecture Challenges for Data Intensive Computing. IEEE, pp. 4–6. doi:10.1109/WICSA.2008.50.

Grinev, M., Grineva, M., Hentschel, M., Kossmann, D., 2011. Analytics for the real-time web. Proc. VLDB Endow. 4, 1391–1394.

Gulisano, V., Jimenez-Peris, R., Patino-Martinez, M., Valduriez, P., 2010. StreamCloud: A Large Scale Data Streaming System. IEEE, pp. 126–137. doi:10.1109/ICDCS.2010.72.

Hausenblas, M., 2014. Twitter Open-Sources its MapReduce Streaming Framework Summingbird [Online]. URL https://www.infoq.com/news/2014/01/twitter-summingbird (accessed 10.28.16).

Gronlund, C.J., 2016. What is Hadoop on HDInsight? [Online]. URL https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-introduction (accessed 11.26.16).

Hirschheim, R., 1985. Information Systems Epistemology: An Historical Perspective. Research Methods in Information Systems 13–35.

Java Platform SE 8 [WWW Document], 2014. . Java Platform, Standard Edition 8. URL https://docs.oracle.com/javase/8/docs/api/ (accessed 10.30.16).

Kaplan, B., Maxwell, J., 2005. Qualitative Research Methods for Evaluating Computer Information Systems, in: Anderson, J., Aydin, C. (Eds.), Evaluating the Organizational Impact of Healthcare Information Systems. Springer New York, pp. 30–55.

Khare, S., An, K., Gokhale, A., Tambe, S., Meena, A., 2015. Reactive Stream Processing for Data-centric Publish/Subscribe, in: Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, DEBS ’15. ACM, New York, NY, USA, pp. 234–245. doi:10.1145/2675743.2771880.

Kreps, J., 2014. Questioning the Lambda Architecture - O’Reilly Media [Online]. URL https://www.oreilly.com/ideas/questioning-the-lambda-architecture (accessed 10.28.16).

Krishnan, S., Tse, E., 2013. Hadoop Platform as a Service in the Cloud [Online]. Netflix Tech Blog. URL http://techblog.netflix.com/2013/01/hadoop-platform-as-service-in-cloud.html (accessed 10.30.16).

Kuhn, T.S., 1964. The structure of scientific revolutions. University of Chicago Press, Chicago.

Land, F. (1992) The Information Systems Domain. In: Information Systems Research, Issues, Methods and Practical Guidelines. Great Britain, Blackwell Scientific Publications, pp.6–13.

Lederman, A., 2016. Let's take a look at some really BIG "big data" [Online]. URL http://www.deepwebtech.com/2016/06/lets-take-a-look-at-some-really-big-big-data/ (accessed 10.30.16).

Lee, A.S., Briggs, R., Dennis, A.R., 2014. Crafting Theory to Satisfy the Requirements of Explanation. IEEE, pp. 4599–4608. doi:10.1109/HICSS.2014.566.

Leedy, P.D. and Ormrod, J.E. (2000) Practical Research: Planning and Design. 7 edition. Upper Saddle River, N.J, Pearson.

Loesing, S., Hentschel, M., Kraska, T., Kossmann, D., 2012. Stormy: an elastic and highly available streaming service in the cloud. ACM Press, p. 55. doi:10.1145/2320765.2320789.

Marz, N., 2014. History of Apache Storm and lessons learned - thoughts from the red planet - thoughts from the red planet [Online]. URL http://nathanmarz.com/blog/history-of-apache-storm-and-lessons-learned.html (accessed 10.28.16).

Marz, N., Warren, J., 2015. Big Data: Principles and best practices of scalable realtime data systems, 1st ed. Manning Publications.

McClean, J., 2015. Plumbing Java 8 Streams with Queues, Topics and Signals [Online]. Medium. URL https://medium.com/@johnmcclean/plumbing-java-8-streams-with-queues-topics-and-signals-d9a71eafbbcc (accessed 10.27.16).

Miller, M., 2015. Innovate or Die: The Rise of Microservices. The Wall Street Journal.

Miller, S., 2014. Collaborative Approaches Needed to Close the Big Data Skills Gap. Journal of Organization Design 3, 26. doi:10.7146/jod.9823.

Mohammed, A.F., Humbe, V.T., Chowhan, S.S., 2016. A review of big data environment and its related technologies. IEEE, pp. 1–5. doi:10.1109/ICICES.2016.7518904.

Nadareishvili, I., Mitra, R., McLarty, M. & Amundsen, M. (2016) Microservice Architecture: Aligning Principles, Practices, and Culture. 1 edition. O'Reilly Media.

Neumeyer, L., Robbins, B., Nair, A., Kesari, A., 2010. S4: Distributed Stream Computing Platform. IEEE, pp. 170–177. doi:10.1109/ICDMW.2010.172.

Newman, S. (2015) Building Microservices. 1 edition. Beijing Sebastopol, CA, O'Reilly Media.

Oates, B.J. (2005) Researching Information Systems and Computing. 1 edition. London; Thousand Oaks, Calif, SAGE Publications Ltd.

O'Malley, O., 2008. Apache Hadoop Wins Terabyte Sort Benchmark | hadoopnew - Yahoo [Online]. URL https://developer.yahoo.com/blogs/hadoop/apache-hadoop-wins-terabyte-sort-benchmark-408.html (accessed 10.30.16).

Park, K., Nguyen, M.C. & Won, H. (2015) Web-based collaborative big data analytics on big data as a service platform. In: 2015 17th International Conference on Advanced Communication Technology (ICACT). pp.564–567.

Pattinson (2012) Measuring Data Centre Efficiency. Leeds: Leeds Metropolitan University.

Pcollection [Online], 2016. URL https://cloud.google.com/dataflow/model/pcollection (accessed 11.26.16).

Peng, D., Dabek, F., 2010. Large-scale Incremental Processing Using Distributed Transactions and Notifications, in: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI'10. USENIX Association, Berkeley, CA, USA, pp. 251–264.

Ramnarayan, J., Menon, S., Wale, S., Bhanawat, H., 2016. SnappyData: a hybrid system for transactions, analytics, and streaming: demo. ACM Press, pp. 372–373. doi:10.1145/2933267.2933295.

Rosemann, M., Vessey, I., 2008. Toward Improving the Relevance of Information Systems Research to Practice: The Role of Applicability Checks. MIS Q. 32, 1–22.

Roth, W.D. and Mehta, J.D. (2002) The Rashomon Effect: Combining Positivist and Interpretivist Approaches in the Analysis of Contested Events. Sociological Methods & Research, 31 (2), pp.131–173.

Schmidt, E., 2015. Google Cloud Platform Blog: Announcing General Availability of Google Cloud Dataflow and Cloud Pub/Sub [Online]. URL https://cloudplatform.googleblog.com/2015/08/Announcing-General-Availability-of-Google-Cloud-Dataflow-and-Cloud-Pub-Sub.html (accessed 11.26.16).

Shah, S., 2016. What are the different components available with an HDInsight cluster? [WWW Document]. URL https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-component-versioning#hdinsight-standard-and-hdinsight-premium (accessed 11.26.16).

Simmonds, R.M., Watson, P., Halliday, J., Missier, P., 2014. A Platform for Analysing Stream and Historic Data with Efficient and Scalable Design Patterns. IEEE, pp. 174–181. doi:10.1109/SERVICES.2014.40.

Soley, R., Stone, J. and Castaldini, F. (2016) IDG Quick Pulse Results Webcast: CIO Survey: IIoT Adoption -The Real Barriers & Opportunities Ahead.

Sommerville, I., 2010. Software Engineering, 9 edition. ed. Pearson, Boston.

Stream Analytics – Real-time data analytics [Online], 2016. URL https://azure.microsoft.com/en-gb/services/stream-analytics/ (accessed 11.26.16).

Tejedor, R.C., 2013. Lambdoop, a framework for easy development of big data applications.

Land, F. (1992) The Information Systems Domain. In: Information Systems Research, Issues, Methods and Practical Guidelines. Great Britain, Blackwell Scientific Publications, pp.6–13.

Taylor, R.N., Medvidovic, N., Dashofy, E., 2009. Software Architecture: Foundations, Theory, and Practice, 1 edition. ed. John Wiley & Sons, Hoboken, NJ.

Thomas, G., 2013. How to Do Your Research Project, 2nd Revised edition edition. ed. Sage Publications Ltd, Los Angeles.

The Netflix Tech Blog: Evolution of the Netflix Data Pipeline [Online], 2016. URL http://techblog.netflix.com/2016/02/evolution-of-netflix-data-pipeline.html (accessed 10.30.16).

The Zettabyte Era—Trends and Analysis - Cisco [Online], 2016. Zettabyte Era—Trends Anal. URL http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html (accessed 10.30.16).

Tichy, W.F., Lukowicz, P., Prechelt, L., Heinz, E.A., 1995. Experimental evaluation in computer science: A quantitative study. Journal of Systems and Software 28, 9–18. doi:10.1016/0164-1212(94)00111-Y

Vambenepe, W., 2015. Google Cloud Platform Blog: Announcing Google Cloud Dataflow runner for Apache Flink [Online]. URL https://cloudplatform.googleblog.com/2015/03/announcing-Google-Cloud-Dataflow-runner-for-Apache-Flink.html (accessed 11.26.16).

Villari, M., Celesti, A., Fazio, M., Puliafito, A., 2014. AllJoyn Lambda: An architecture for the management of smart environments in IoT. IEEE, pp. 9–14. doi:10.1109/SMARTCOMP-W.2014.7046676.

What is Amazon EMR? [Online], 2016. URL http://docs.aws.amazon.com/ElasticMapReduce/latest/ManagementGuide/emr-what-is-emr.html (accessed 11.26.16).

What is Google Cloud Dataproc? [Online], 2016. URL https://cloud.google.com/dataproc/docs/concepts/overview (accessed 11.26.16).

Wills, J., 2015. New in Cloudera Labs: Google Cloud Dataflow on Apache Spark - Cloudera Engineering Blog [Online]. URL http://blog.cloudera.com/blog/2015/01/new-in-cloudera-labs-google-cloud-dataflow-on-apache-spark/ (accessed 11.26.16).

Zikopoulos, P., deRoos, D., Parasuraman, K., Deutsch, T., Giles, J. & Corrigan, D. (2013) Harness the Power of Big Data The IBM Big Data Platform. 1 edition. McGraw-Hill Education.